License Plate Matching Using Neural Networks

Kelvyn SOSOO (GMU) David OUYANG (CSUST) Mengjun WANG (CSUST) Mentors: Lee HAN (UTK) & Kwai WONG (UTK)

Background

- License Plate Recognition (LPR) technology is used to gather vehicle location data
- Location Data includes instances of Amber Alerts, Toll Roads Speed/Travel Time, etc.
- The License Plate Matching (LPM) method incorporated includes a 97% match rate of vehicles, and a 60% read accuracy
- Programs Used: Python, Matlab

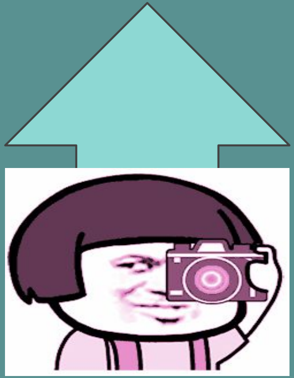
GOAL: Raise the 60% by using Image Processing. Find a new measure to matching plate by using supervised learning.

How It Works

Capture images → Image recognition → Plate matching

R 1

LP



Procedure

Screen the License Plate images

Image Processing to segment every Character

Neural network training

Matching two string

Image Processing

Step 1 : Manipulation of Data

00022507_A_f_2017-05-08_08-09-45.825_p02_H605307_3_9...

	A	B
1	2010-05-27	06:08:15.200000
2	2010-05-27	06:57:52.700000
3	2010-05-27	08:35:40.520000
4	2010-05-27	09:04:17.330000
5	2010-05-27	09:13:15.730000
6	2010-05-27	12:30:27.910000
7	2010-05-27	14:52:51.240000
8	2010-05-27	14:59:15.240000
9	2010-05-27	15:00:35.960000
10	2010-05-27	15:01:10.170000
11	2010-05-27	15:12:58.100000
12	2010-05-27	15:13:56.770000
13	2010-05-27	15:16:17.660000
14	2010-05-27	15:40:27.030000
15	2010-05-27	15:56:24.700000
16		

Step 2: Image binarization

```
ret, imgf = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
fig.add_subplot(2, 2, 1)
plt.imshow(imgf, cmap = 'gray')
cv2.imwrite("thresh{}.jpg".format(i),  imgf)            #write ev
P1 = cv2.imread("thresh{}.jpg".format(i))
grayscaleimg = cv2.cvtColor(P1, cv2.COLOR_BGR2GRAY)
```

Original

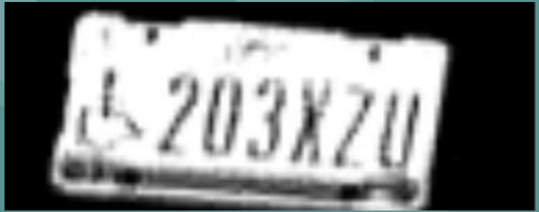
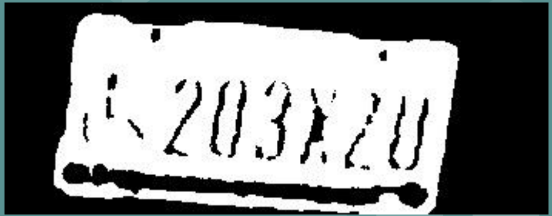


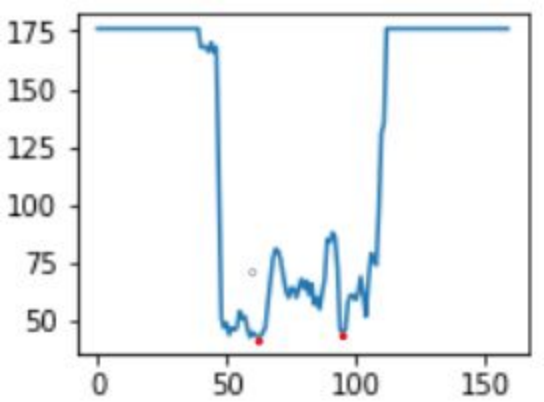
Image enhancement

Midterm

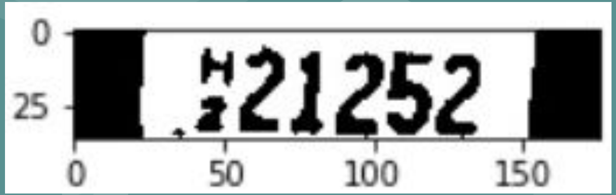


Final

Step 3 : Read the Number of Black Pixels Vertically



[176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 168, 168, 168, 166,
170, 166, 168, 108, 51, 47, 49, 44, 47, 46, 47, 54, 51, 52,
47, 43, 45, 44, 43, **43**, 45, 47, 58, 67, 77, 81, 80, 76, 69,
63, 60, 64, 64, 60, 64, 68, 64, 67, 61, 66, 57, 60, 55, 62,
68, 85, 84, 88, 86, 73, 46, **45**, 46, 58, 61, 61, 59, 62, 69,
61, 52, 68, 79, 76, 74, 98, 131, 135, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176]



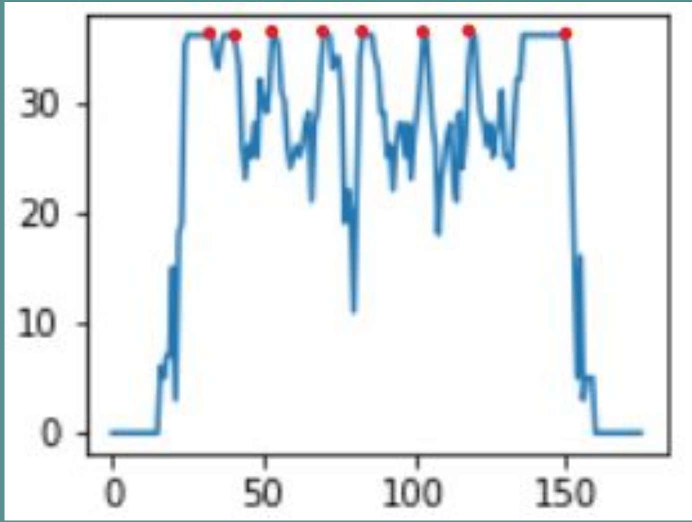
np.argmin(row_nz[0 : floor(len(row_nz)/2)]) == 59 row_nz(59) = 43

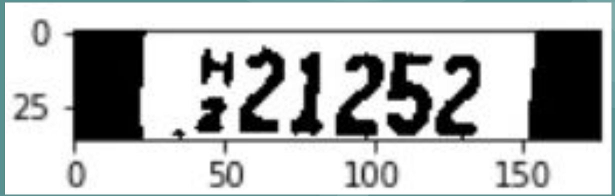
np.argmin(row_nz[floor(len(row_nz)/2) :]) == 95 row_nz(95) = 45

Two key points coordinate: (59, 43) (95,45)

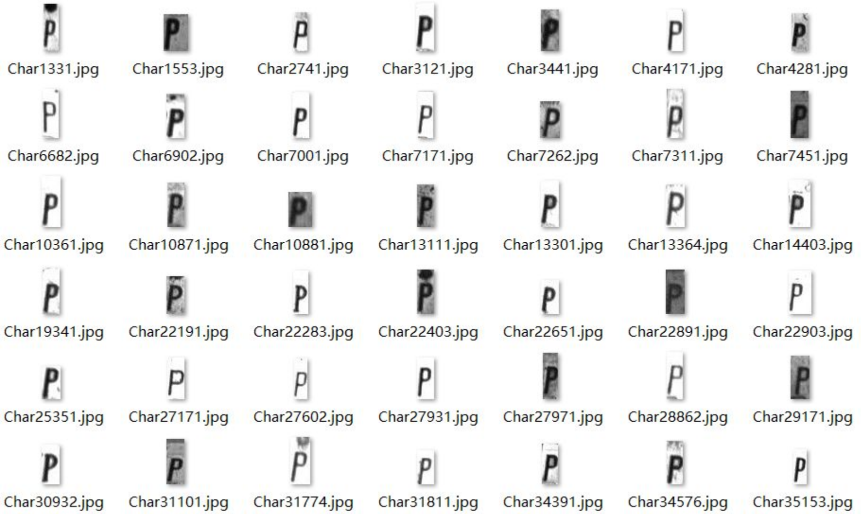
Step 4 : Read the Number of White Pixels Horizontally

KEY POINT (CUT POINT) : [33, 40, 54, 72, 86, 104, 120, 150]





Outcome


Char1331.jpg Char1553.jpg Char2741.jpg Char3121.jpg Char3441.jpg Char4171.jpg Char4281.jpg
Char6682.jpg Char6902.jpg Char7001.jpg Char7171.jpg Char7262.jpg Char7311.jpg Char7451.jpg
Char10361.jpg Char10871.jpg Char10881.jpg Char13111.jpg Char13301.jpg Char13364.jpg Char14403.jpg
Char19341.jpg Char22191.jpg Char22283.jpg Char22403.jpg Char22651.jpg Char22891.jpg Char22903.jpg
Char25351.jpg Char27171.jpg Char27602.jpg Char27931.jpg Char27971.jpg Char28862.jpg Char29171.jpg
Char30932.jpg Char31101.jpg Char31774.jpg Char31811.jpg Char34391.jpg Char34576.jpg Char35153.jpg


Char2271.jpg Char2544.jpg Char2582.jpg Char2691.jpg Char2981.jpg Char2982.jpg
Char11941.jpg Char12431.jpg Char12481.jpg Char12741.jpg Char12771.jpg Char13481.jpg
Char16851.jpg Char17021.jpg Char17531.jpg Char18171.jpg Char18451.jpg Char18521.jpg
Char20183.jpg Char20251.jpg Char20391.jpg Char21081.jpg Char21292.jpg Char21861.jpg
Char25121.jpg Char25231.jpg Char25391.jpg Char25411.jpg Char25601.jpg Char26263.jpg
Char31391.jpg Char31392.jpg Char31411.jpg Char32242.jpg Char32341.jpg Char32391.jpg

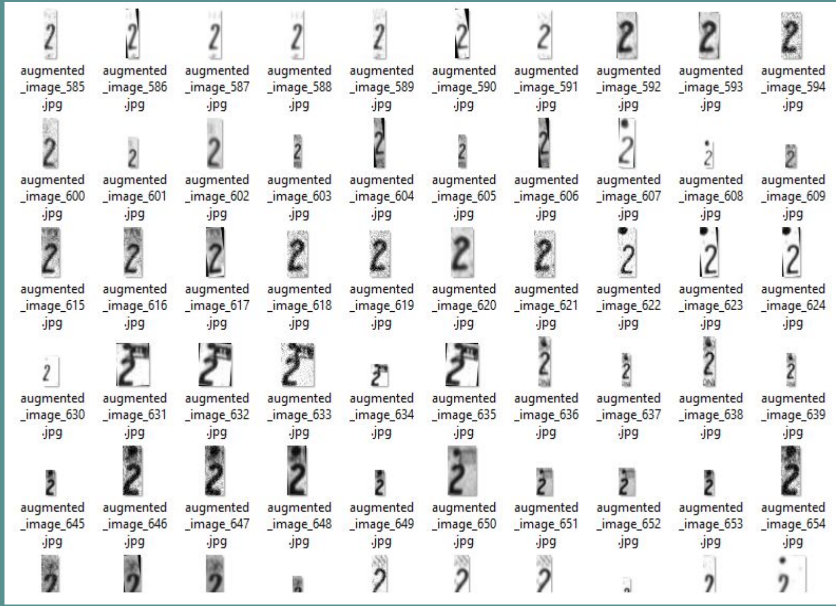

1.jpg 2.jpg 3.jpg 4.jpg 5.jpg 6.jpg 7.jpg

Supervised Learning: Neural Network

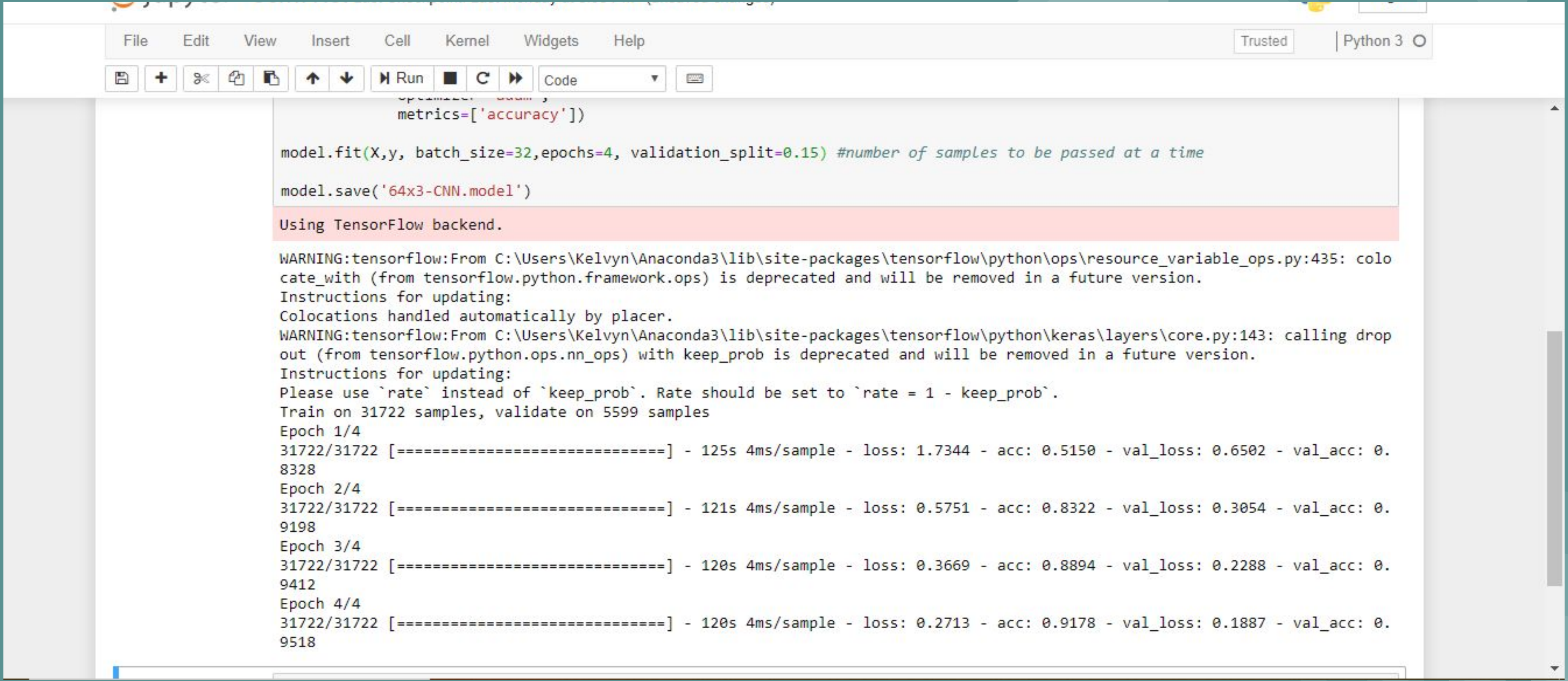
- Previous slide presented the outcome of Character Segmentation
    - It is very time consuming to transfer the characters to the proper label/category
- Instead of spending countless hours manually moving files, Data Augmentation was implemented
- Categories included A-Z and 0-9

Attempts

- Two different training datasets were tested: Grayscale and Binary Images



Midterm Performance




```
             metrics=['accuracy'])

model.fit(X,y, batch_size=32,epochs=4, validation_split=0.15) #number of samples to be passed at a time

model.save('64x3-CNN.model')
```

```
Using TensorFlow backend.
```

```
WARNING:tensorflow:From C:\Users\Kelvyn\Anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colo
cate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\Kelvyn\Anaconda3\lib\site-packages\tensorflow\python\keras\layers\core.py:143: calling drop
out (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
Train on 31722 samples, validate on 5599 samples
Epoch 1/4
31722/31722 [==============================] - 125s 4ms/sample - loss: 1.7344 - acc: 0.5150 - val_loss: 0.6502 - val_acc: 0.
8328
Epoch 2/4
31722/31722 [==============================] - 121s 4ms/sample - loss: 0.5751 - acc: 0.8322 - val_loss: 0.3054 - val_acc: 0.
9198
Epoch 3/4
31722/31722 [==============================] - 120s 4ms/sample - loss: 0.3669 - acc: 0.8894 - val_loss: 0.2288 - val_acc: 0.
9412
Epoch 4/4
31722/31722 [==============================] - 120s 4ms/sample - loss: 0.2713 - acc: 0.9178 - val_loss: 0.1887 - val_acc: 0.
9518
```

- After four epochs, the model was able to reach a validation accuracy of 95.18%

Final Performance

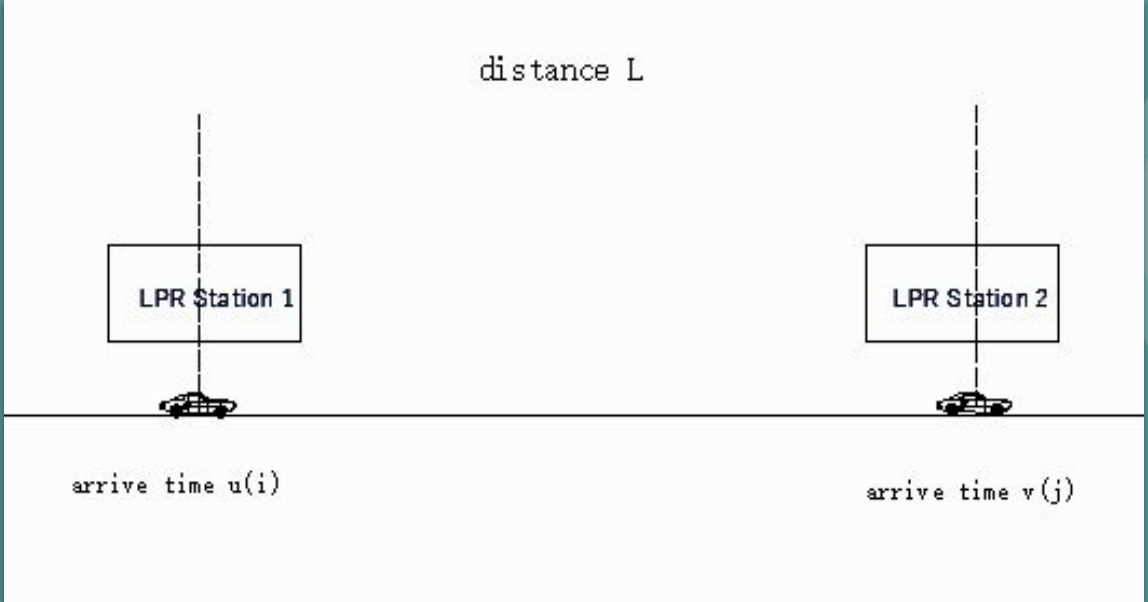
```
Train on 31723 samples, validate on 5599 samples
Epoch 1/3
31723/31723 [==============================] - 153s 5ms/sample - loss: 1.1023 - acc: 0.6947 - val_loss: 0.1719 - val_acc: 0.
9489
Epoch 2/3
31723/31723 [==============================] - 169s 5ms/sample - loss: 0.2075 - acc: 0.9367 - val_loss: 0.0764 - val_acc: 0.
9791
Epoch 3/3
31723/31723 [==============================] - 177s 6ms/sample - loss: 0.1232 - acc: 0.9608 - val_loss: 0.0580 - val_acc: 0.
9812
```

- After three epochs, the model was able to reach a validation accuracy of 98.12%

Model Usage

- Characters from seperate folders/ license are identified
- Stored as strings in csv file

Plate Matching



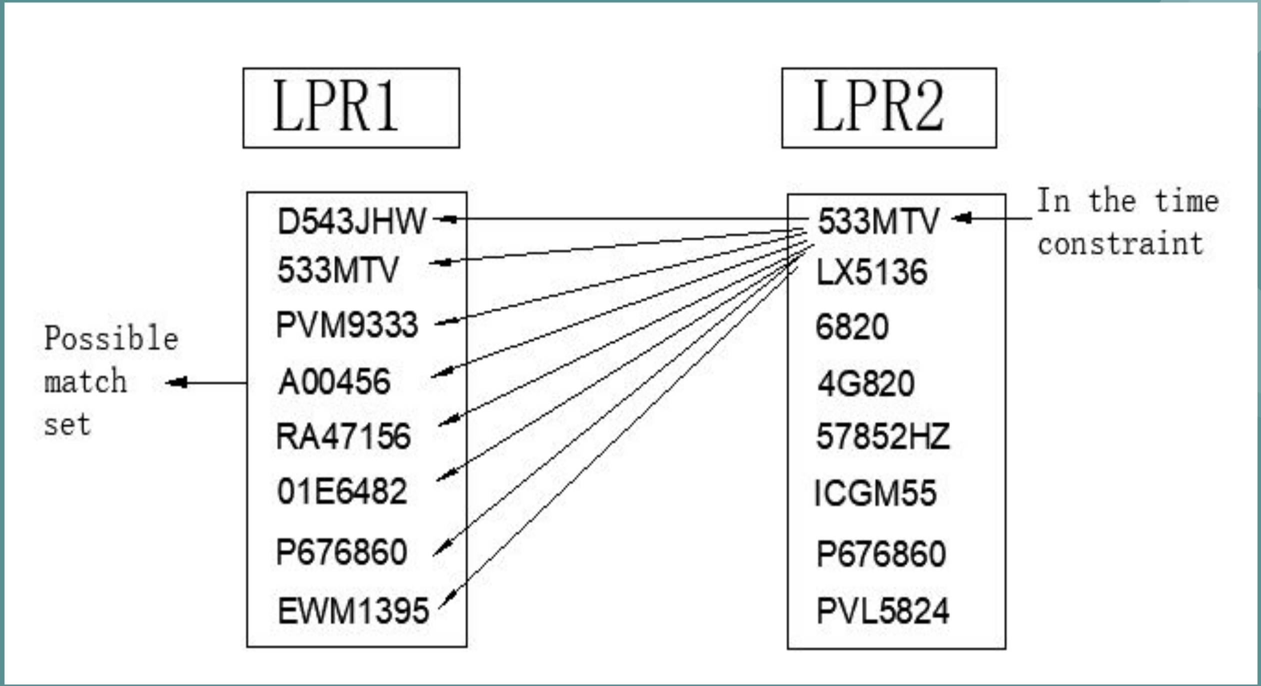
1. v(j), u(i) are both the arrive time.
2. max, min are the speed of passing LPR stations.
3. The distance between two stations is L.

TIME CONSTRAINT

$$\frac{L}{max} \le v(j) - u(i) \le \frac{L}{min}$$

Goal: To judge whether different plate characters are from the same car

Self-learning



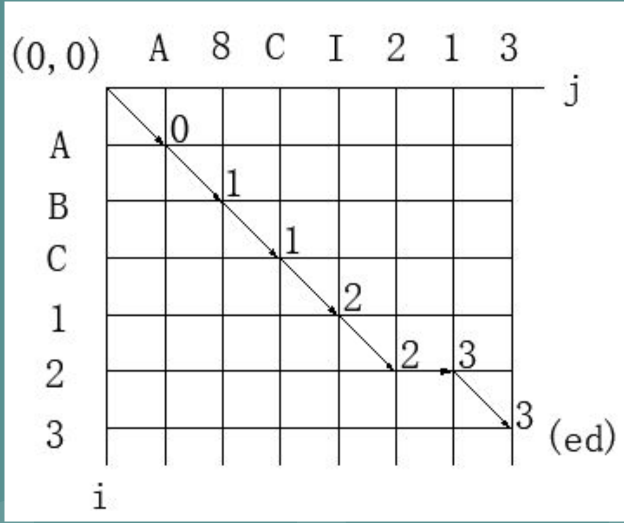
The candidate set

1. Use the time constraints to find all possible plates matches.
2. Put all these selected plates into a set named candidate set 'S', every string in the set named S(i).
3. Get several pairs of plates. Look for the smallest edit distance required to transform each other,
4. Choose the one which shows up firstly.

Character-transition Matrix

For example, there are two plate strings.
A8CI213 & ABC123

B → 8 substitution
1 → I substitution
1 → - deletion



The edit distance between two different license plates and the edit paths on grids.

A-A
B-8
C-C
1-I
2-2
3-3

(1) Find every pair of possible match.
(2) Calculate the edit distance path.
(3) Find all the ***associated characters***.
(4) Calculate the Character-transition matrix.
(5) Iterating and updating the matrix until it is not change.

Association Matrix

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	-
0																																					
1																			1																		
2			1																																		
3				1																																	
4																																					
5																																					
6																																					
7																																					
8																																					
9																																					
A											1																										
B									1																												
C													1																								
D																																					
E																																					
F																																					
G																																					
H																																					
I																																					
J																																					
K																																					
L																																					
M																																					
N																																					
O																																					
P																																					
Q																																					
R																																					
S																																					
T																																					
U																																					
V																																					
W																																					
X																																					
Y																																					
Z																																					
-																																					

The initial character-transition matrix.

Self-learning:
By iterating to calculate the transforming probability between different characters.

$$p(b|a) = \rho_{ab}/\rho_a$$

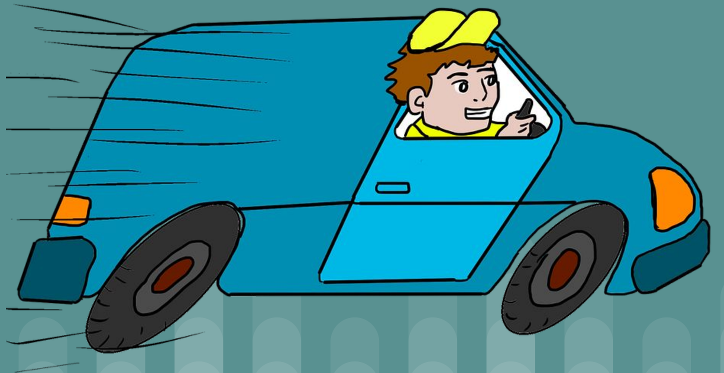
Pab is the value of every grid in the Character-transition matrix.
a is the sum of every row in the Character-transition matrix.

Obtain an **association matrix** by calculating the *conditional probability.*

Final Association Matrix

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	-
0	78	0	0	0	0	0	0	0	1	0	0	0	0	3	0		0	0	0	0		0	0		11	0	0	0	0	0	2	0	0		0		2
1	0	82	0	1	1	0	0	2	0	0	0	0		0	0	0	0	0	8	2	0	0	0	0	0	0			0	1	0	0	0	0	1	0	0
2	0	0	91	0	0	0	0	1	0	0	0	0		0	0		0		0	0	0	0			0	1		0	0	0	0	0	0	0		5	1
3	0	1	0	91	0	1	1	1	0	1	0	0	0	0	0		0	0	0	2		0	0		0	0		0	0	0	0	0	0		0	0	1
4	0	1	0	0	88	1	2	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	1
5	0	0	0	0	1	91	2	0	0	1	0	0	0	0	0	0	0		0	0		0			0	0	0	0	3	0	0	0			0		1
6	0	1	0	1	1	1	82	0	3	0	1	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	1	1	1	0	0	0	93	0	0	0	0	0	0	0		0	0	0	0		0	0		0	1		0	0	0	0	0	0	0	0	1	1
8	2	1	0	0	0	0	1	0	82	1	0	5	0	0	0		0	1	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0			0	1
9	1	1	0	1	1	0	0	1	1	91	0	1	0	0	0		0	0	0	0		0	0		0	0	0	0	1	0	0	1		0	0		0
A	0	3	1	0	3	1	2	0	3	1	74	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0		1	0	1	1	0	0	2		1	2
B	2	3	0	0	0	0	0	0	29	2	0	53	1	0	0	0	0	1	3	1		0	0		0	1	0	1	0	0	0	0	0		0	0	0
C	0	1			3		0	0	1	0	0	0	92		0	1	0	0	0	0			0					0									1
D	17	3	0	1	0	0	0	0	0	0	0	0	0	61	0			0	1	0		0	1	0	10	1	0	0		0	3		0		0	0	1
E	3	1	0	0	0		2	0	0	0	2	0	0	0	76	2	0		4	0		0			0	4	0		0	0	0	0			0		2
F		0	0		0		2	2	0	0	0				2	83		2	2	0			0			5							0	0	0	2	0
G	0	1	0	0	0	2	22	2	5	0	0	2	2	0	0		59	2	1	0		0	0			3		0	0	0	0	0	0		0	0	0
H		0					0	0	0		0				0			94	2	0		0	1	0	0		0	0			0		1				
I	1	33	0	2	0	0	0	2	1	2	0	0	0	2	0	1	0	1	38	2		2	1	0	0	1	0	0	0	4	2	1	0	0	2	1	0
J	0	16	1	4	0	1	0	0	0	2	0	0		0	0	0	0	1	3	65		1	0	1	0	0		0	0	0	1	0	0		0	0	2
K		2					2				0							0	0		86		1	0				3					2	4			0
L	0	8	0	0	2		0	0	0	0	0	0		0	4		0		16	0		68							0	2	0	0				0	0
M	0	1	0	0	0		2	0	0	0	0	0		3	0		0	9	1	0	6	0	52	11	0			2	0	0	0	0	11	0			0
N		0	0	3		0	0	3	0	0	0	0		0			0	5	0	0			3	79						0	3		3		0	0	0
O	62	3	0	0	0		0	0	1	0	0	0	0	4	0		0	0	1	0		0	0		21		3	0	0	0	3	0				0	2
P	0	1	1	0	0	0	0	0	0	1	0	0		0	1		0	0	1	0		0	0		0	92		0	0	0	0	0			1	0	2
Q	36	3	0	1	1	10	0	0	0	11	0	0	0	3	0	0	0		3	1		0			13		11		1	0	3	0					1
R	0	1	0	0	0		0	0	8	0	2	0		0	0		0	0	1	0	5	0	0	2				75	0	0	0	0	0	2			3
S	0	4	1	3	0	28	1	0	3	3	1	0	1	0	0	0	1	0	1	0		0	0		0	0		0	50	0	0	0			0	0	2
T	0	13	0	3	0	2	0	5	0	3	0	0	0	0	2	0	2		11	1		2			0	0			0	49	0	0			2	0	3
U	7	0			0		0	0	0		0		0	3		1		1	19	0			0		3		0				65		0				0
V	0	2	0	0	0	0	0	0	0	3	0	0		0	0		0	1	1	0		0	0		0	0			0	2	2	79	2		5		2
W	0	1	0	2	0	0	0	0	0	0	2	0		2	0		0	9	1	0		2	5	6	0			0	0	0	0	0	68	0	0		0
X		0	0	2	0	0	2	2	0	0	0	0	0	0			0		3	0	3					0		0		0			3	86	0	0	0
Y		1	0				0	3			0			0			0	0	0		0					1				2		7	1	1	83	0	0
Z		0	10	0	2		0	4		0	0			0			0		1	0						0		0	0	0				1		83	0
-	12	4	6	4	3	8	2	4	8	7	1	0	3	0	1	1	0	5	2	0	1	2	1	0	2	6	0	1	1	2	0	2	2	1	6	2	1

This is a 37 by 37 matrix.
0-9 & A-Z & SPACE
The x axis is LPR 1 reading.
The y axis is LPR 2 reading.
The value of every grid is the conditional probability of two characters being misread at two sites.



Matching

For instance, there are two pairs of license plates:

44S5H2 4455HZ

4415HZ 4455HZ

Which is the match one??

$$d(x \to y) = min\{\sum_{k=0}^{n} log(\frac{1}{p(i_k, j_k)})\}$$

$d(x \to y)$ is the cost of transforming x to y.

x_i	y_j	$p(y_j\|x_i)$	$\log\left(\frac{1}{p(y_j\|x_i)}\right)$
"4"	"4"	0.885	0.122
"4"	"4"	0.885	0.122
"S"	"5"	0.280	1.273
"5"	"5"	0.914	0.090
"H"	"H"	0.937	0.065
"2"	"Z"	0.055	2.906
$GED(x \to y) = \sum \log\left(\frac{1}{p(y_j\|x_i)}\right) =$			4.579
z_i	y_j	$p(y_j\|z_i)$	$\log\left(\frac{1}{p(y_j\|z_i)}\right)$
"4"	"4"	0.885	0.122
"4"	"4"	0.885	0.122
"1"	"5"	0.001	6.535
"5"	"5"	0.914	0.090
"H"	"H"	0.937	0.065
"Z"	"Z"	0.829	0.188
$GED(z \to y) = \sum \log\left(\frac{1}{p(y_j\|z_i)}\right) =$			7.122

The minimum one is the match one.

44S5H2 & 4455HZ

Matching with FuzzyWuzzy

- Based on Fuzzy Logic / Levenshtein Distance formula
- Simple and fast way of string matching

Future Works

- Improving efficiency of MATLAB matching code
- Improve character segmentation
- Find fully autonomous implementation of license plate matching

THANKS FOR LISTENING, ANY QUESTIONS?